

TITLE OF THE INVENTION

IPV6 HEADER RECEIVING APPARATUS
AND IPV6 HEADER PROCESSING METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the priority of Korean Patent Application No. 2003-2086, filed on January 13, 2003, in the Korean Intellectual Property Office, the disclosure of which is incorporated herein in its entirety by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0002] The present invention relates to an apparatus and method of receiving and processing an IP header, and more particularly, to an IPv6 header receiving apparatus capable of being implemented in hardware and an IPv6 header processing method.

2. Description of the Related Art

[0003] To solve the growing shortage of 32-bit Internet Protocol (IP) addresses caused by the rapid increase in networking and IP nodes, the Internet Engineering Task Force (IETF) started developing an IPv6 protocol to replace an IPv4 protocol and developed an IPv6 protocol in the early 1960s. An IPv6 protocol was designed by adding new features to the IPv4 protocol, based on the experiences accumulated from the design of the IPv4 protocol.

[0004] Most parts of the IPv6 protocol are implemented in software. In other words, in conventional software implementations of the IP protocol, a packet received from a lower layer is copied and stored in memory and then processed. In the conventional software implementations of the IP protocol, data transmission between layers is achieved with a memory buffer chaining method using a pointer.

[0005] FIG. 1 shows the structure of a conventional IPv6 input module. When a MAC controller 120, which processes a data link layer and a physical layer, collects data

corresponding to one frame, the MAC controller 120 uses a CRC checksum to determine whether the frame is defective. If the frame is not defective, the MAC controller 120 transmits a packet via a dedicated interrupt terminal to a machine 110. A processor 111 in the machine 110 receives data from a memory 113 using a pointer 112, processes the received data, and overwrites a location in the memory 113 with the processed data. In this way, packet processing in each layer is achieved through memory overwrites.

[0006] However, in such a software implementation of IP, memory to store a packet is indispensable, and accordingly, extra time to access the memory is required. The extra time represents a significant amount of latency with respect to a packet, and consequently, generates a considerable overhead in a TCP/IP-ported machine.

[0007] FIG. 2 shows the format of an extended IPv6 header. An IPv6 header includes a version field for a 4-bit IP version number; an 8-bit traffic class field; a 20-bit flow label field; a payload length field, which represents the length of an IPv6 payload and is composed of an unsigned 16-bit integer; a next header field, which is a 8-bit selector that identifies the type of an extended header following the IPv6 header; an 8-bit hop limit field; a 128-bit origin address field representing the source of a packet; and a 128-bit destination address field representing the destination of the packet.

[0008] An IPv6 packet may have no extended headers, one extended header, or a plurality of extended headers. Each extended header is identified by a next header field included in the previous basic or extended header. Each extended header has a length that is an 8-octet multiple.

[0009] The extended IPv6 packet shown in FIG. 2 includes 6 extended headers. If a plurality of extended headers are used in one packet, a hop-by-hop option extended header, a destination option extended header, a routing extended header, a fragment extended header, an authentication extended header, and a destination option extended header, the plurality of extended headers can be sequentially arranged between an IPv6 header and an upper layer header (not shown).

[0010] The hop-by-hop option extended header is used to transport selective information that is inspected by each node along a packet transport path.

[0011] The next header field in the hop-by-hop option extended header is an 8-bit selector and identifies the type of header immediately following the hop-by-hop option extended header. An extended header length field in the hop-by-hop option extended header is expressed with an unsigned 8-bit integer and represents the length of the hop-by-hop option extended header that is formed in units of 8 octets. The field next to the extended header length field is a variable length field, which includes an option. The next header field and the extended header length field are commonly applied to all extended headers.

[0012] The routing extended header is used to list a plurality of intermediate nodes that a packet must visit when transported from an IPv6 source to a destination.

[0013] The fragment extended header is used when a packet has a size larger than the maximum transport unit (MTU) and the packet must be transported from an IPv6 source to a destination.

[0014] The destination option extended header is used to transport selective information that is inspected by the destination node specified in the packet.

[0015] FIGS. 3A through 3C show the header structures of IPv6 implemented with various link layers. An IPv6 header over Ethernet, as shown in FIG. 3A, includes a destination Ethernet address, a source Ethernet address, and an IPv6 header & payload. An IPv6 header over a Fiber Distributed Data Interface (FDDI), as shown in FIG. 3B, includes a destination FDDI address, a source FDDI address, and an IPv6 header & payload.

[0016] An IPv6 header over a token ring, as shown in FIG. 3C, includes a destination address, a source address, and an IPv6 header & payload. As observed from FIGS. 3A through 3C, IPv6 headers are processed in units of 8 octets regardless of the type of link layer implementation.

[0017] A conventional IPv6 receiving apparatus has increased latency due to the frequency of required memory accesses. When a conventional IPv6 receiving apparatus is implemented in hardware, additional space is required to predict and count the variable extended header length.

[0018] In Korean Patent Publication No. 2002-34230, filed on June 19, 2002 and published on September 5, 2002, entitled “IPv6 implementing apparatus and a physical medium interface

unit, an IPv6 header processing unit, and an upper layer interface unit which are used in the IPv6 implementing apparatus", IPv6 and ICMPv6 protocols, which are normally processed by software or the operating system (OS), are implemented in hardware in real time without a loss in communication speed. In the IPv6 protocol implementing apparatus, the physical medium interface unit is connected to a physical interface layer. The IPv6 header processing unit processes an IPv6 basic header and IPv6 extended headers and is connected to the physical medium interface unit. The upper layer interface unit is connected to an upper layer protocol, such as, TCP or UDP.

[0019] Korean Patent Publication No. 2002-34230 discloses an IPv6 implementing apparatus which is implemented in hardware. However, a structure in which an IPv6 header is received and stored is not specified.

SUMMARY OF THE INVENTION

[0020] Various aspects and/or advantages of the invention will be set forth in part in the description which follows and, in part, will be obvious from the description, or may be learned by practice of the invention.

[0021] In accordance with an aspect of the present invention, there is provided an IPv6 header receiving apparatus capable of processing an IPv6 header in real time without wasting memory and an IPv6 header processing method.

[0022] According to an aspect of the present invention, there is provided an IPv6 header receiving apparatus including a register with a data size that is a multiple of an octet, which receives IPv6 header data in units of an octet, stores the IPv6 header data, and transmits the stored IPv6 header data to a module for processing an IPv6 header that corresponds to the IPv6 header data, and modules for an IPv6 basic header or various types of IPv6 extended headers, which receive the IPv6 header data from the register and process the IPv6 header data.

[0023] According to an aspect of the present invention, there is also provided an IPv6 header receiving apparatus comprising a counter, which counts up to an amount of IPv6 extended header data that can be transmitted in units of an octet at a time, a register with a data size that is a multiple of an octet, which receives IPv6 header data in units of an octet, stores the IPv6

header data, and, if the counter has completed counting up to the specified amount, transmits the stored IPv6 header data to a module for processing an IPv6 header that corresponds to the IPv6 header data, and modules for an IPv6 basic header or various types of IPv6 extended headers, which receive the IPv6 header data from the register and process the IPv6 header data.

[0024] According to an aspect of the present invention, there is also provided an IPv6 header receiving apparatus comprising an octet indicator, a register with a data size that is a multiple of an octet, a controller, and modules for an IPv6 basic header or various types of IPv6 extended headers. The octet indicator counts up to an amount of IPv6 extended header data that can be transmitted in units of an octet at a time. The register receives IPv6 header data in units of 8 octets and stores the IPv6 header data. The control unit analyzes the IPv6 header data stored in the register to determine a type and length corresponding to the IPv6 header data. If the octet indicator has completed counting up to the specified amount, the control unit causes transmission of the IPv6 header data with the determined length to a module for processing an IP header that corresponds to the IPv6 header type. The modules for an IPv6 basic header or various types of IPv6 extended headers receive the IPv6 header data from the register and process the IPv6 header data.

[0025] According to another aspect of the present invention, there is provided an IPv6 header processing method which includes filling a register with IPv6 header data received in units of a predetermined size, which is a multiple of an octet, identifying an IPv6 header type by analyzing the IPv6 header data filled in the register, and transmitting the IPv6 header data to a module corresponding to the identified IPv6 header type.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] These and/or other aspects and advantages of the invention will become apparent and more readily appreciated from the following description of the embodiments, taken in conjunction with the accompanying drawings of which:

FIG. 1 shows an example of the structure of a conventional IPv6 input module;

FIG. 2 shows the format of an extended IPv6 header;

FIGS. 3A through 3C show the header structures of IPv6 implemented with various link layers;

FIG. 4 is a block diagram of an IPv6 header receiving apparatus according to an embodiment of the present invention;

FIG. 5 shows the detailed structure of the next header status register of FIG. 4; and

FIG. 6 is a flowchart for illustrating an IPv6 header processing method according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0027] Reference will now be made in detail to the embodiments of the present invention, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to the like elements throughout. The embodiments are described below to explain the present invention by referring to the figures.

[0028] Because processing of an IPv6 header format is based on 8 octets, an IPv6 implementing apparatus according to the present invention is implemented in hardware using an extra device that can monitor 8 octets.

[0029] Referring to FIG. 4, an IPv6 header receiving apparatus according to an embodiment of the present invention includes an IP receiving module 400 and a register file 450. The IP receiving module 400 receives an IPv6 header in units of 8 octets and transports the received 8 octets of data to the register file 450. The register file 450 receives and processes the IPv6 header.

[0030] The IP receiving module 400 includes a control unit 410, a temporary register 420, a shift register 430, and an octet indicator 440. The temporary register 420 receives data, the amount of which depends on the type of interface, from a link layer 460 and stores the received data. For example, if a PCMCIA interface is used, the size of the temporary register 420 would be 2 octets.

[0031] The control unit 410 analyzes data immediately after the data is stored in the temporary register 420, issues a transport command for transmitting data to the shift register 430, and informs the octet indicator 440 of the data amount to be dynamically received.

[0032] To be more specific, the control unit 410 includes a header analyzer 411, a next header status register 412, a length register 413, and a path determiner 414.

[0033] The header analyzer 411 analyzes data received from the temporary register 420, ascertains the type and length of a header corresponding to the data, and sets the next header type and next header length in the next header status register 412 and the length register 413, respectively. To be more specific, an extended IPv6 header as shown in FIG. 2 includes an 8-bit next header field and an 8-bit extended header length field. The type of the next extended header following a current extended header can be ascertained from the next header field, and the length of the next extended header can be ascertained from the extended header length field. These analysis results are set in the next header status register 412 and the length register 413.

[0034] The next header status register 412 stores data representing the types of existing extended headers. The detailed structure of the next header status register 412 is shown in FIG. 5.

[0035] Referring to FIG. 5, the next header status register 412 includes a destination option header (DOH) bit (a destination node bit), an encapsulating security payload (ESP) bit, an authentication header (AH) bit, a fragment header (FH) bit, a routing header (RH) bit, a DOH bit (an intermediate node bit), a hop-by-hop header (HBH) bit, and a reserved bit. If an extended header exists, the corresponding bit is set to true. When packet processing is concluded, each bit is set to false.

[0036] Such a next header status register provides the field information necessary for checksum calculation on a network layer and provides a control signal that is used when an ICMP message is generated, due to an error in the header field or a problem in the networking media.

[0037] The length register 413 stores data representing the length of an extended header.

[0038] The path determiner 414 uses information stored in the next header status register 412 and the length register 413 to determine which module of the register file 450 receives the data stored in the shift register 430. Once the path determiner 414 has also determined the amount of data that should be transmitted, it instructs the shift register 430 to transmit the determined amount of data to the determined module of the register file 450.

[0039] The shift register 430 receives two octets of data at a time from the temporary register 420 and waits until 8 octets have been accumulated. When 8 octets of data have been accumulated, the shift register 430 shifts the 8-octet data to the register file 450 based on the determined information from the path determiner 414.

[0040] The octet indicator 440 receives count amount corresponding to an octet of data and continues to count until the shift register 430 receives 8 octets of data.

[0041] An auxiliary counter 441 counts up to 8 octets of data, corresponding to the amount stored in the shift register 430 and transmitted at one time to the register file 450. A main counter 442 counts the maximum effective length of each extended header. Also, the main counter 442 receives information, corresponding to the extended header length, from the header analyzer 411 and assists the shift register 430 in receiving and transmitting the correct amount of data corresponding to the extended header length to a module of the register file 450, which is capable of processing the header of the corresponding data.

[0042] The register file 450 includes a basic header module 451, a routing header module 452, an AH module 453, an ESP module 454, a DOH module 455, an HBH module 456, and an upper layer module 457.

[0043] Each of the modules of the register file 450 receives data from the shift register 430 in predetermined units. When the contents of a header have been completely received, each module processes the header.

[0044] FIG. 6 is a flowchart for illustrating an IPv6 header processing method according to the present invention. First, in operation S610, the IP receiving module 400 waits for data transmission by the link layer 460.

[0045] In operation S620, a MAC layer transmits data in predetermined units to the IP receiving module 400.

[0046] In operation S630, the IP receiving module 400 stores the data in the temporary register 420. For example, the received data can be stored in units of 2 octets.

[0047] In operation S640, the header analyzer 411 of the control unit 410 receives and analyzes data stored in the temporary register 420 and causes the path determiner 414 to

determine a path for data in the shift register 430. In other words, the header analyzer 411 can ascertain the type and length of an extended header next to a current header. The type of the next extended header is stored in the next header status register 412, and the length of the next extended header is stored in the length register 413. Also, the header analyzer 411 transmits information about the length of each extended header to the main counter 442 and informs the shift register 430 of the amount of data to be transmitted to each header module. The information stored in the next header status register 412 and the length register 413 are used when the path determiner 414 determines which module receives the data stored in the shift register 430.

[0048] In operation S650, the header data stored in the temporary register 420 is transported to the shift register 430.

[0049] In operation S660, it is determined whether the auxiliary counter 441 has been terminated. For example, if the size of the temporary register 420 is 2 octets, and the size of the shift register 430 is 8 octets, the auxiliary counter 441 would be terminated when the count has reached 4.

[0050] If it is determined in operation S660 that the auxiliary counter 441 has not yet been terminated, the method goes back to operation S630, in which data received from the MAC layer is stored in the temporary register 420.

[0051] If it is determined in operation S660 that the auxiliary counter 441 has been terminated, then in operation S670 the path determiner 414 instructs the shift register 430 to transmit data of 8 octets to a module of the register file 450.

[0052] In operation S680, it is determined whether the value of the main counter 442 has exceeded a predetermined value. Because the main counter 442 maintains information representing the length of the next extended header, the main counter 442 helps the shift register 430 transport all the data constituting the next extended header to a module capable of processing the next extended header.

[0053] If it is determined in operation S680 that the main counter value has not exceeded the predetermined value, that is, the length of the next extended header, the method goes back to operation S620, in which the MAC layer transmits data in a predetermined unit.

[0054] If it is determined in operation S680 that the main counter value has exceeded the predetermined value, a next packet is received, in operation S690. If the main counter value has exceeded the predetermined value, that is, if the actual amount of data received is greater than the amount of data specified as the length of the next extended header, it may be determined that an error has occurred in the extended header. In this case, instead of receiving the extended header following the current extended header, the packet corresponding to the current extended header is not used, and the next packet is received and processed.

[0055] As described above, an IPv6 receiving apparatus according to the present invention can be implemented in hardware without the need of installing a memory device. Thus, data can be processed in real time, and the data processing time, the memory capacity, and the manufacturing costs of the IPv6 receiving apparatus are reduced.

[0056] Although a few embodiments of the present invention have been shown and described, it would be appreciated by those skilled in the art that changes may be made in this embodiment without departing from the principles and spirit of the invention, the scope of which is defined in the claims and their equivalents.